{EPITECH}

JAVA SEMINAR_

< DAY 02 - OBJECT ORIENTED PROGRAMMING />

JAVA SEMINAR

Let's make some **O**bject **O**riented **P**rogramming.
To help you with your research, here are today's concepts (in no particular order):

- ✓ Classes definition ;
- ✓ Objects and instantiations ;
- ✓ Constructors ;
- ✓ Methods and attributes ;
- ✓ New operator ;
- ✓ Methods and attributes visibility ;
- ✓ Optional parameters.

The article Object Oriented Programming Explained with Memes may help your understanding while entertaining you.

Exercise 01

Delivery: `./ex_01/Gecko.java`

Create a new `Gecko` class.
Every time a new Gecko is created, `Hello!` followed by a newline must be displayed.

Here is a test example:

```java
public class Example {
    public static void main(String[] args) {
        Gecko arthur = new Gecko();
        Gecko benjy = new Gecko();
    }
}
```

```
Terminal
$> java Example
Hello!
Hello!
```

Exercise 02

Delivery: `./ex_02/Gecko.java`

Copy your `Gecko.java` from exercise 01 and add a new constructor that takes a name as parameter.

If a name is specified, `Hello <Name>!` followed by a newline must be displayed.
Otherwise, the constructor is called with `Unknown` as value of the `name` attribute.

This parameter should be stored in a public `name` attribute.

Here is a test example:

```java
public class Example {
    public static void main(String[] args) {
        Gecko arthur = new Gecko("Arthur");
        Gecko benjy = new Gecko();

        System.out.println(arthur.name);
        System.out.println(benjy.name);
    }
}
```

Terminal

```
$> java Example | cat -e
Hello Arthur!$
Hello!$
Arthur$
Unknown$
```

Exercise 03

Delivery: `./ex_03/Gecko.java`

From the `Gecko.java` from the previous exercise, have the `name` attribute not to be public anymore.

You have to find out for yourself what it should be.

However, so that the name is available outside the object, you need to create your very first method!
It must be called `getName` and return... the Gecko's name!

Here is a test example:

```java
public class Example {
    public static void main(String[] args) {
        Gecko arthur = new Gecko("Arthur");
        Gecko benjy = new Gecko();

        System.out.println(arthur.getName());
        System.out.println(benjy.getName());
    }
}
```

Terminal

```
$> java Example | cat -e
Hello Arthur!$
Hello!$
Arthur$
Unknown$
```

Exercise 04

Delivery: `./ex_04/Gecko.java`

Copy your `Gecko.java` from the previous exercise and add an `age` attribute to your Gecko class.
It should be possible to set it as a second parameter during the construction of the object.

> Previous constructor rules still apply.

This attribute must have its own getter and setter, respectively `getAge` and `setAge`.

Also add a new `status` method to your Gecko.
It takes no parameters and displays a sentence according to the Gecko's age.

> You must use a `switch` statement, the `if` keyword is ot allowed.

The method must display the following sentences:

- ✓ `Unborn Gecko`, if the age is 0 ;
- ✓ `Baby Gecko`, if the age is 1 or 2 ;
- ✓ `Adult Gecko`, if the age is between 3 and 10 ;
- ✓ `Old Gecko`, if the age is between 11 and 13 ;
- ✓ `Impossible Gecko`, otherwise.

> Each of these sentences must be followed by a newline.

Exercise 05

Delivery: `./ex_05/Gecko.java`

It is time to give the gift of gabbing to our Geckos!
Copy your previous `Gecko.java` and add a new public method, called `hello`.

When called with a string, it must display `Hello <string>, I'm <Name>!`:

✓ with `<string>` being the string given as parameter ;
✓ and `<Name>` being the name of the Gecko.

However, if an integer is given as parameter, it must display `Hello, I'm <Name>!` as often as the number given as parameter.

Every messages must be followed by a newline.

In all other cases, the method does nothing.

```java
public class Example {
    public static void main(String[] args) {
        Gecko mimi = new Gecko("mimi");
        mimi.hello("Titi");
        mimi.hello(2);
    }
}
```

Terminal

```
$> java Example | cat -e
Hello mimi!$
Hello Titi, I'm mimi!$
Hello, I'm mimi!$
Hello, I'm mimi!$
```

Exercise 06

Delivery: `./ex_06/Gecko.java`

Copy your `Gecko.java` from the previous exercise and add an `eat` method which:

- ✓ takes a string as parameter ;
- ✓ is case insensitive ;
- ✓ returns nothing.

Depending on the argument (value of the parameter), the Gecko must display:

- ✓ `Yummy!`, if the argument is equal to `Meat` ;
- ✓ `Erk!`, if the argument is equal to `Vegetable` ;
- ✓ `I can't eat this!`, otherwise.

As usual, every sentence must be followed by a newline.

Moreover, add an `energy` attribute to our Gecko:

- ✓ by default it is equal to `100` ;
- ✓ a gecko's energy should always be between 0 and 100 (included).

Add a setter (`setEnergy`) and a getter (`getEnergy`) for this attribute.

Every time our Gecko eats something, it will win or lose some energy. If it eats:

- ✓ `Meat`, it will win 10 energy ;
- ✓ `Vegetable`, it will lose 10 energy (a Gecko is carnivorous) ;
- ✓ in all other cases, his energy will not be modified.

Exercise 07

Delivery: `./ex_07/Gecko.java`

Copy your `Gecko.java` from the previous exercise and implement a `work` method that takes no parameter and eturns nothing.
When the method is called, it must display:

- ✓ `I'm working T.T`, if the Gecko has at least 25 energy. Then the energy will decrease by 9 (it is working 8 hours a day so it needs enough energy to work the whole day).
- ✓ `Heyyy I'm too sleepy, better take a nap!`, if the Gecko has 24 or less energy. Then it will give your Gecko back 50 energy.

As usual, every sentence will be followed by a newline.

Exercise 08

Delivery: `./ex_08/Gecko.java`, `./ex_08/Snake.java`

Let's implement a new `fraternize` method that takes one parameter.
If the parameter is a Gecko Object, our Gecko will be happy and go drink with his friend.
It will cost both of them 30 Energy.
They will both say (starting with the current Gecko) `I'm going to drink with <otherName>!`

If one of them doesn't have enough energy:

- ✓ it must display `Sorry <otherName>, I'm too tired to go out tonight.` ;
- ✓ the other will then display `Oh! That's too bad, another time then!`.

If both of them are too tired to go out, they will both display `Not today!`.

If the parameter is a `Snake` and if Gecko's energy is:

- ✓ greater than or equal to 10, the Gecko displays `LET'S RUN AWAY!!!` and it's energy is set to 0.
- ✓ less than 10, the the Gecko plays dead and displays ...

Feel free to add any method that you reckon necessary.

v 3.5.1

{EPITECH}